**ReadMe**

Here is our source link for BPlustree and Rtree:

<https://github.com/AhmadElsagheer/DBMS/tree/master/SagSolheeriman-A1>

**Intro:**

- What we do is storing every node in a single page (serialization and deserialization of nodes).

- Handling duplicates in overflow pages.

-some other methods to help us in the DPApp.

**Demonstration:**

Here is an explanation for the added methods.

**updates In BTree** :

1.Implementation of a search range method and not equal method using complete and complete 2 as helper methods

lines: A.138 to 148 B.149 to 152

**updates In BTreeNode:**

1. In the constructor, the node is being serialized after being initialized.

lines: A.32 B.36 to 40.

2.adding refresh method which tends to serialize any node after any action that makes any change to the node.

lines: A.163 to 171.

**updates In BTreeInnerNode:**

1. Replacing the array of children(of type BTreeNode) with an array of filenames of children(of type String) to make sure that deserializing a node gives only one node and its children should be deserialized to be in memory.

lines: A.19 B.31 and every line of the code that contain children array.

2.getChild method deserializes the page of a specific filename to get the child node.

lines: A.42 to 51.

3.setChild method sets the filename of the child node then calls the refresh method to serialize the current node.

line: A.61,62.

4.getFirstChild & getLastChild methods, deserialize the page of the first & last child node .

lines: A.73 to 78 ; B.89 to 93

5. Insert method, deserializes Btreenode of specific index and calls refresh method.

lines: A.119 to 123 ; B.125,127,132,134,136,143

6. The split method uses the refresh method to serialize current node.

lines: A.161

7.The InsertAt method uses the refresh method to serialize current node

lines: A.212

8.Delete method, deserializes Btreenode of specific index

lines: A.255 to 259; B.264 to 268

9. Merge Method, calls refresh method after any update to the node.

lines: A.400

10.deleteAt method, calls refresh method

lines: A.425

11.adding searchn method

lines: A.471 to 479

**updates In BTreeLeafNode:**

1. Replacing the array of records(of type reference) with an array of record filenames of records references(of type string), and replacing the next pointer node to a next leaf node filename.

lines: A.16,17,24

2.getNext method deserializing the next leaf node

lines: A.32 to 41

3.setNext method, setting next leaf node filename and calling refresh method

lines: A.52,55 B.53,56

4. Instead of getrecord and setrecord methods which returns references, we implemented getRecordfilename and setRecordfilename which returns a string and calls refresh method

lines: A.76 to 79 B.95 to 99

5. Instead of getFirstRecord and getLastRecord which return reference, we implemented getFirstRecordfilename and getLastRecordfilename which return string

lines: A.122 to 124 B.133 to 135

6.insertAt method calling refresh method

lines: A.221

7.split method calling refresh method

lines: A.253

8.deleteAt method calling refresh method

lines: A.371,387

9.borrow method calling refresh method

lines: A.421,442

10.merge method calling refresh method

lines: A.470,493

11.implementing our own complete and complete2 methods to help us in range search and notequal method.

lines: A.561 to 637 B.639 to 639

12.implementing our searchd method to help in finding the appropriate place for inserting a key and implementing getLastone as a helper method.

lines: A.520 to 550 B.502 to 519

for Duplicates Handling:

we implemented the idea of overflow pages through:

1.implementing class BTreeDupPage which represents pages of references that each key in the leaf node will point to.

2.doing both serialization and deserialization of DupPage.

3.implementing getDupPage in BTreeLeafNode that deserialize specific DupPage and using it in needed methods

lines: A.277 to 288

4.implementing all needed methods for handling duplicates.